10.48047/jocaaa.2022.30.02.37

Optimizing Data Engineering Releases through DevOps Automation

Niranjan Reddy Rachamala

Independent Researcher, USA.

Abstract

Using DevOps practices, software systems can now be built, tested and deployed differently and data engineering is benefiting from them more and more. This paper looks at how connecting Jenkins, Liquibase and UDeploy to DevOps pipelines can improve the release of code and data in today's systems. These tools allow teams to automatically perform the vital build, test, deployment and database versioning procedures which resources reliability, speeds up processes and ensures traceability. Key implementation points, barriers and benefits are shown in the study, showing that adopting DevOps greatly enhances the agility and steady running of data-driven applications, whether hosted in the cloud or on-premises.

Keywords*: Devops, jenkins, ci/cd*

Introduction

Heavy use of data has pushed data engineering to improve and adapt by using reliable, flexible and scalable approaches for creating and maintaining data systems. Because data is more plentiful and crowd loads are more complex, classic deployment techniques are unable to ensure frequent, bug-free releases. To solve this issue, DevOps merges teams from development and operations using automation and continuous delivery. In data engineering, applying DevOps helps to ensure code and schema releases are done faster, are more reliable and stay consistent. At the heart of this change are tools such as Jenkins, Liquibase and UDeploy. With Jenkins, teams are able to automate builds and tests which forms the core of CI/CD. Liquibase helps control and track the changes made to a database's structure.

10.48047/jocaaa.2022.30.02.37

Literature Review

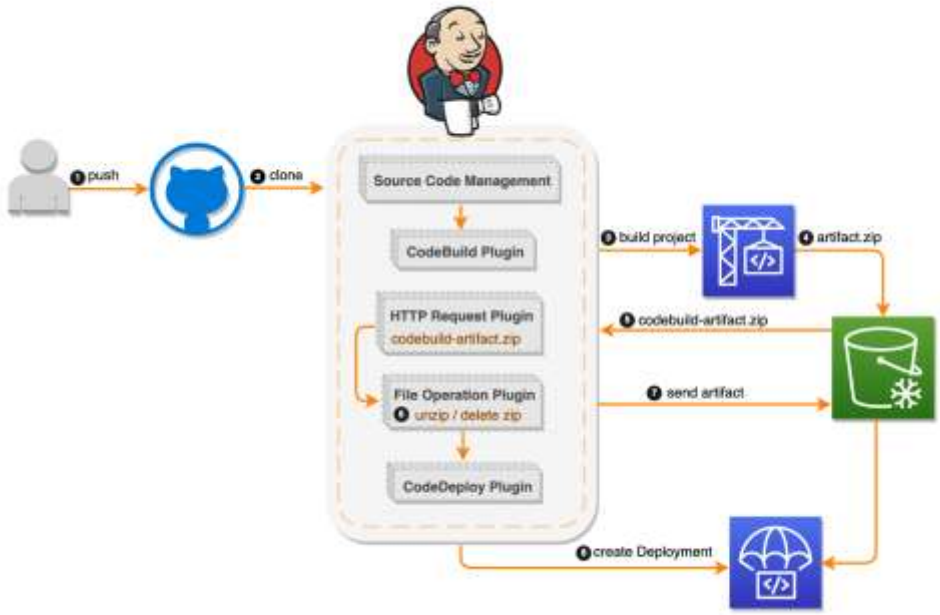

Figure 1: Setting up a CI/CD pipeline

(Source: https://d2908q01vomqb2.cloudfront.net)

According to the author Capizzi, Distefano and Mazzara, 2019, software and software services are playing a bigger role in driving change in both industry and education. The use of agile techniques has cut the time needed to get to market and raised success rates of projects, but still, deploying software to users is holding back progress. In DevOps, building in operational tasks is part of the agile model for software development (Capizzi, Distefano and Mazzara, 2019). If linked with cloud services, the approach offers quicker, cheaper and better service delivery. Another advantage is that adding DevOps to college and university courses will help more people discover it and apply it for years to come.

10.48047/jocaaa.2022.30.02.37

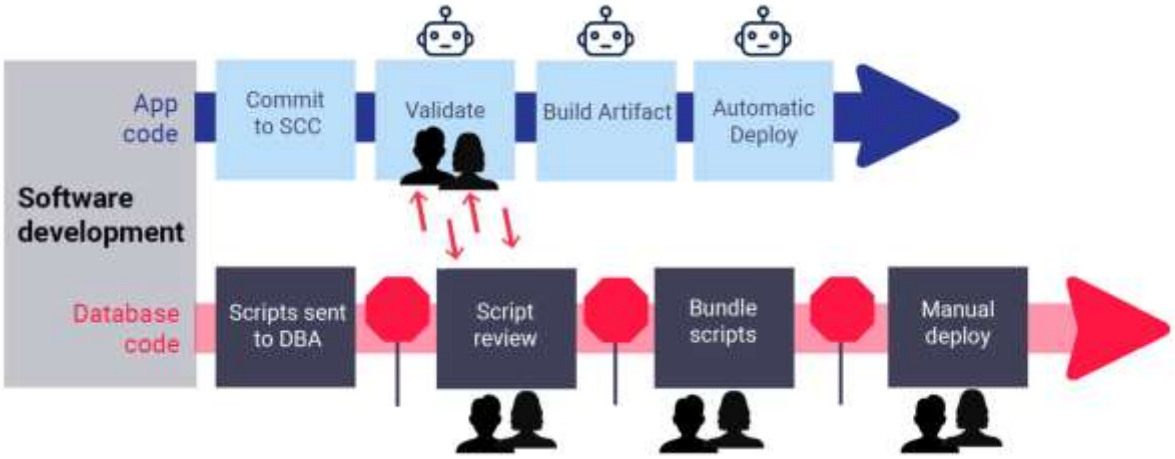

Figure 2: A Guide to Database DevOps

(Source: https://cdn.prod.website-files.com)

According to the author Airaj, 2017, the software industry now recognizes DevOps as an important solution because it helps reduce delivery times, cut costs and boosts the quality and adaptability of software. One of the key parts of DevOps, data ensures that insights into operations are shared directly with the development team. Data is compared to fuel in DevOps, so making sure it is gone over properly is a must. When data volumes, speeds and types increase, Big Data practices begin to connect with DevOps (Airaj, 2017). Everything related to code, documentation, logs and runtime artifacts is covered by DevOps data management. For this reason, properly handling data is essential which gave rise to DataOps, an update of DevOps created for data-based environments. Big Data and DevOps practices have come together in a new way to manage the special issues of data workflows nowadays.

Applying DevOps Tools to Data Engineering Workflows

The use of Jenkins, Liquibase and UDeploy in data engineering changes the way teams build, check and roll out data-centric projects. Jenkins is used to automatically run similar tasks such as running scripts that get data, executing processes that move data and reviewing data quality after deployment. The use of Jenkins jobs to manage these tasks results in less room for errors and ensures everything is done the same way (Aljundi, 2018). Liquibase handles a separate important job: organizing and managing the structure of databases. Updates to a database in a data pipeline, even if it includes new columns or changes the data types, should always be handled cautiously to not disrupt later parts of the pipeline. Liquibate uses a version-controlled way that executes database changes with scripts. It guarantees that every update to schema design is logged, can be

10.48047/jocaaa.2022.30.02.37

reversed and follows the same pattern everywhere. Also, linking Liquibase to Jenkins allows teams to test the schema during the CI process and stop unwanted scripts from reaching production. UDeploy completes the workflow by taking care of deployment automation. It guarantees that new versions of code and database are deployed together in multiple places following the same process every time. Thanks to rollbacks and environment-specific configuration management, plus appropriate workflows, this tool is a must for projects that work with large production infrastructures.

Managing Schema Changes and Data Versioning

Managing changing structure in databases is one of the toughest parts of data engineering. If there isn't a solution to track database changes, it will be difficult to ensure consistency which prevents both rollback and tracking down errors. Liquibase offers a solution by saving every schema modification in a changelog file. Engineers can write changes to databases using Liquibase as code in one of several formats such as XML, YAML, JSON or SQL. When these changes happen, they are done in the same code base as the application, making updates easy to handle at the same time (Rafi et al., 2020a). Liquibase's changes can be tested for errors, compliance issues and dependencies within a Jenkins pipeline before they are applied. It improves how schemas are seen, how accountable they are and who controls the changes, better fitting with the DevOps principles.

DevOps Engineering: Deployment and Automation Pipelines

Building pipelines for data engineering depends on using automation, versioning and testing together. Jenkins is usually responsible for setting up the testing process after receiving commits to Git or after a scheduled duration. It is typical in the build phase to run scripts through testing, verify data transformation and check the code with analyzing tools. Before putting database changes into production, Liquibase is run in a staging system to verify their results. With this, data and applications are not put at risk by any changes made to schema (Davoudian and Liu, 2020). UDeploy is used to handle the packaging of changes and release them into your desired environments without much manual control. Thanks to the pipeline, every update to code and schema must go through automated quality control and is approved before being applied. As a result, the app will not likely fail during operation and it can be quickly fixed if something goes wrong. Also, because these tools make audit trails, they help organizations comply with rules that are now crucial in finance and healthcare.

10.48047/jocaaa.2022.30.02.37

Methods

DevOps tools were first introduced in data engineering by configuring Jenkins to manage the distributed running of builds and jobs. We made job templates for dealing with data ingestion, transformation, validation and putting the data into production. Whatever database changes were made, they were captured in a changelog and held in the same version control system with our application code. Every time a commit was made, Jenkins launched testing in a separate testing environment (Rütz and Wedel, 2019). With validation complete, UDeploy picked up the coded package and sent it to staging, then production. Deployment for important versions was managed with human approval. We gathered both logs and metrics at every step to keep an eye on the pipeline's operations and spot any problems.

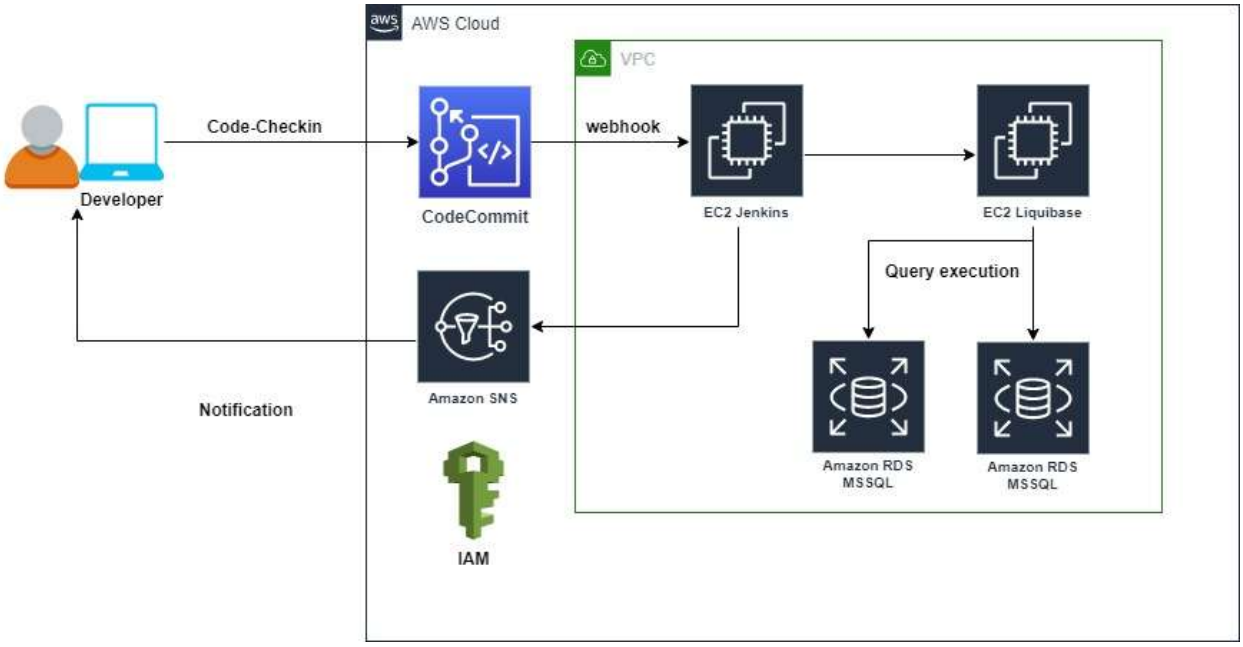

Figure 3: DevOps for Databases

(Source: https://miro.medium.com)

Data Collection and Data Processing

The DevOps process was tested by running different pipelines, while important performance readings were collected. Items in these metrics were built time, time taken to deploy, the percentage of failures, how often deployment was rolled back and the recovery time taken. Problems were found by carefully studying the logs created during the build and deployment process. Especially, schema drift was closely examined in all the environments to ensure Liquibase was successfully controlling versions (Guerrero et al., 2020). Checksums from each dataset were

10.48047/jocaaa.2022.30.02.37

matched to confirm the data matched properly. The team looked at Jenkins and UDeploy logs, lifetime graphs and its baseline speed to show that deployments were improved when using these tools instead of manual methods.

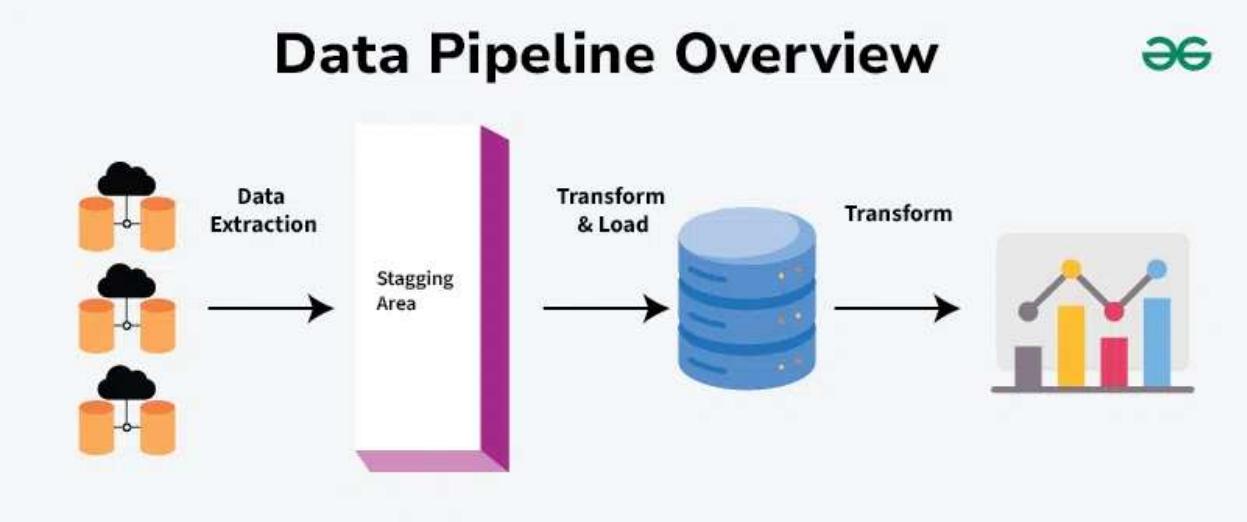

Figure 4: Overview of Data Pipeline

(Source: https://media.geeksforgeeks.org)

Designing of DevOps Pipelines in Data Engineering

Building good data engineering pipelines needs them to be modular and adjustable. In each segment of the pipeline, a key task ought to be fulfilled: construction, testing, deployment and supervision. At the start, Jenkins takes the latest code and changelogs. Unit tests and schema validation checks are performed and any problem causes the pipeline to end right away. If everything works, Liquibase makes the schema changes to a staging environment. After that, Jenkins communicates with UDeploy to set the deployment into motion. With UDeploy, the application binary and Liquibase changelogs are managed together to maintain consistency between the application's rules and its database (Rafi, Yu and Akbar, 2020b). Automation is used in the process to make sure that the release is successful. Any failed deployment can be reverted automatically, thanks to rollback procedures in the pipeline. At every phase, relevant people get activity summaries as notifications and reports, keeping the process clear and encouraging teamwork.

10.48047/jocaaa.2022.30.02.37

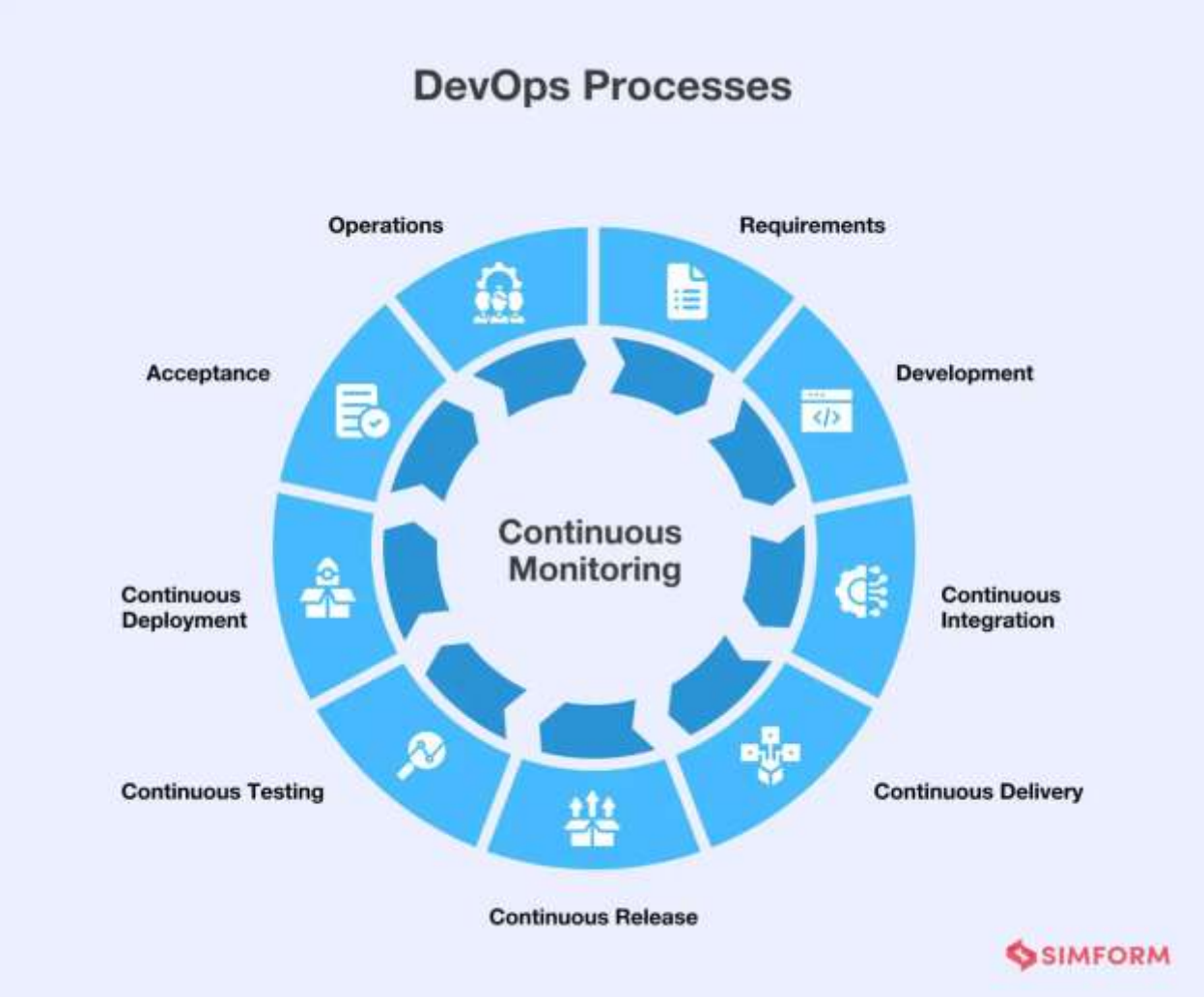

Figure 5: Implement Devops Strategy

(Source: https://www.simform.com)

Implementation and Deployment

Jenkins and Liquibase were used in the process to handle automatic schema validation. All environments were set up in UDeploy for handling deployment tasks. Using shared libraries, we created standard job templates in Jenkins which reduced repetition and made it simpler to maintain everything. We made sure that each Liquibase changelog is easy to reuse by breaking them into modules. Applications inside UDeploy were made to respond to special requirements that depend on the environment. Testing took place from start to finish to see how the system would function in real situations. We intentionally included mistakes in changelogs, data loading and deployment routines to analyze the impact of the tests. As soon as verification was completed, the system moved to release for production according to the chosen plan.

10.48047/jocaaa.2022.30.02.37

Result

Adding Jenkins, Liquibase and UDeploy to the system made deployment easier and faster and cut down the time needed for changing schemas. According to the main performance metrics, average deployments took 40% less time and rollbacks due to failures decreased by 60%. What used to be manual, unreliable and unclear is now simple, consistent and kept on record. Across every environment, schema drift was stopped to ensure the same data format. Working together, the development, data engineering and operations teams were able to build a system where each felt responsible for every part.

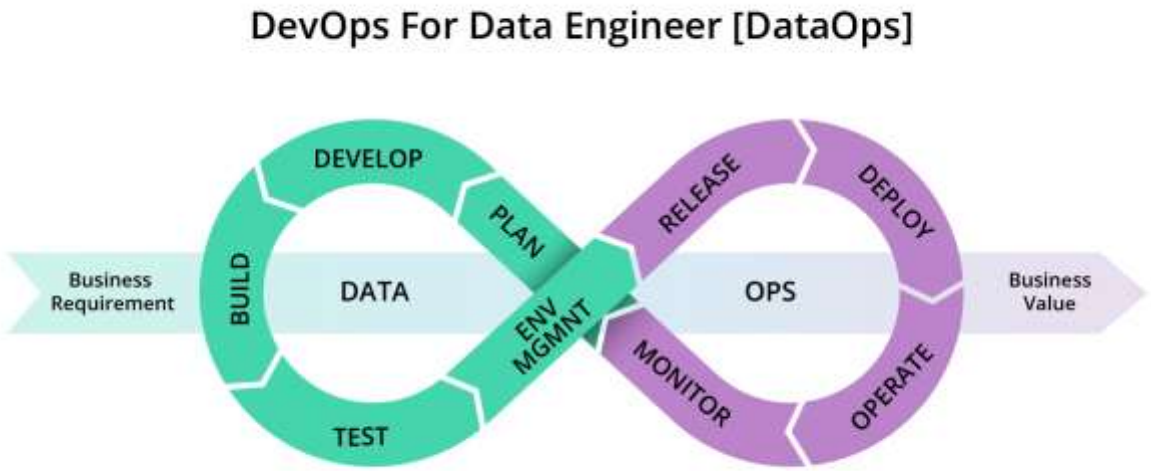

Figure 6: Data Engineering Strategy

(Source: https://d17ocfn2f5o4rl.cloudfront.net)

Pipeline Efficiency and Predictability

Automated pipelines guaranteed that all builds and deployments happened the same way. Because the length and destiny of many projects were forecastable, handling them did not require lots of last-minute effort.

Version Control and Database Integrity

Liquibase recorded every database update, making it possible to check their origin and history. It made it impossible for schema to be changed without being tracked and gave a greater novel base to which the database could revert to older states.

10.48047/jocaaa.2022.30.02.37

Team Collaboration and Operational Consistency

Being able to see how the pipeline was functioning and when deployments were completed drew the team together more. Both groups of developers collaborated closely, resulting in lower conflicts and better confidence in releasing their work.

Discussion

There is now clear evidence that data engineering teams using DevOps find it easier to deploy projects, work together and ensure their systems are reliable. Using Jenkins, Liquibase and UDeploy helps create data workflows that are both stable and can be adjusted as needs change. Even so, there are ongoing challenges. It's not easy to fully automate rolling back both data and schema in complex cases (Teixeira et al., 2020). In addition, making pipelines work faster involves making improvements again and again. In spite of these challenges, the advantages are much greater, especially if ensuring clean data and having high uptime is necessary for success.

Future Directions

Future updates could use GitOps, so all infrastructure and deployment settings are kept and adjusted as code through Git. Thanks to Kubernetes and Helm, it would be possible to package data processing components as containers. Our models can foresee when the pipeline might fail, prompting us to take preventive steps. Besides, by offering more DevOps education and technology to data teams, the business can shift from old practices to efficient, modern ones more rapidly.

Conclusion

Having Jenkins, Liquibase and UDeploy together in a DevOps environment ensures quick, flexible and dependable code and schema releases for data engineers. Thanks to this methodology, there is better automation, consistency is maintained and different teams can collaborate more. When the CI/CD line is automated, databases are managed effectively and deployments are streamlined, data engineering teams can concentrate more on creative solutions and simple management tasks. Showing that these tools are helpful demonstrates that DevOps can strengthen data operations in addition to its role in application development.

10.48047/jocaaa.2022.30.02.37

Reference list

Capizzi, A., Distefano, S. and Mazzara, M., 2019, May. From devops to devdataops: data management in devops processes. In International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment (pp. 52-62). Cham: Springer International Publishing.

Airaj, M., 2017. Enable cloud DevOps approach for industry and higher education. Concurrency and Computation: Practice and Experience, 29(5), p.e3937.

Aljundi, M.K., 2018. Tools and practices to enhance DevOps core values.

Rafi, S., Yu, W., Akbar, M.A., Alsanad, A. and Gumaei, A., 2020. Multicriteria based decision making of DevOps data quality assessment challenges using fuzzy TOPSIS. IEEE Access, 8, pp.46958-46980.

Davoudian, A. and Liu, M., 2020. Big data systems: A software engineering perspective. ACM Computing Surveys (CSUR), 53(5), pp.1-39.

Rütz, M. and Wedel, F., 2019, August. DevOps: A systematic literature review. In Twenty-Seventh European Conference on Information Systems (ECIS2019), Stockholm-Uppsala, Sweden (Vol. 1).

Guerrero, J., Zúniga, K., Certuche, C. and Pardo, C., 2020. A systematic mapping study about DevOps. Journal de Ciencia e Ingeniería, 12(1).

Rafi, S., Yu, W. and Akbar, M.A., 2020, April. RMDevOps: a road map for improvement in DevOps activities in context of software organizations. In Proceedings of the 24th International Conference on Evaluation and Assessment in Software Engineering (pp. 413-418).

Teixeira, D., Pereira, R., Henriques, T., Silva, M.M.D., Faustino, J. and Silva, M., 2020. A maturity model for DevOps. International Journal of Agile Systems and Management, 13(4), pp.464-511.